5. Performance comparison of optimization methods for variational quantum algorithms

5.1. Introduction

Recently we have witnessed an explosion of quantum computer prototypes accessible to researchers in academic and industrial laboratories. Existing quantum hardware has already demonstrated the ability to outperform classical computers in specific mathematically contrived tasks [136, 137]. However, it is still unclear whether noisy intermediate-scale quantum (NISQ) [47] hardware can outperform classical computers on practically useful tasks. Here, variational quantum algorithms (VQA) [39, 40, 87] were introduced as a means of preparing classically-hard quantum states by tuning parameters of a quantum circuit to optimize a cost function by utilising a classical optimizer.

The overall performance of VQAs depends on the performance of the classical optimization algorithm. Finding the limitations of these optimization methods for different VQA tasks is critical if they are to impact research and industry. For this, researchers have proposed new classical optimization algorithms that exploit periodic properties of parametrized quantum circuits [138, 139]. Other works have focused on using machine learning techniques to optimize VQAs [140, 141]. These articles benchmark new optimization techniques relative to standard classical optimizers on a wide variety of systems. However, to the best of our knowledge, no extensive comparison of the most common optimization methods for these tasks has been reported yet.

In this chapter we study three aspects that affect the optimization performance in VQAs. We focus on four off-the-shelf optimizers (SLSQP, COBYLA, CMA-ES and SPSA) for the task of finding an approximate ground-state energy of few physical systems. We first look at two different sampling strategies and how they affect the optimization performance with default optimizer hyper-parameters. Then we focus on hyper-parameter tuning of CMA-ES and SPSA; finding a comparable performance between them given optimal hyper-parameters, with the winner depending on the details of the problem. Finally, we investigate the accuracy of the solutions in the presence of stochastic sampling noise. We define a 'sampling noise floor': a bound on the accuracy that an optimizer can reach when the optimal parameters are those of the best-ever function evaluation. Additionally, we show that CMA-ES algorithm can outperform this 'sampling noise floor' when the optimal parameters are selected from an internal estimate of the optimal candidate. Our main contribution is strong numerical evidence that the optimal parameters of a VQA should not be taken from the best-ever measured function evaluation.

5.2. Background

A variational quantum algorithm attempts to find approximate ground states of an N-qubit quantum system as the output of a circuit $U(\vec{\theta})$ with tunable parameters $\vec{\theta}$. This generates a variational ansatz,

$$|\Psi(\vec{\theta})\rangle \quad = \quad U(\vec{\theta})|\Phi\rangle, \tag{5.1}$$

where the parameters $\vec{\theta} \in [0, 2\pi]^d$ control the rotations of single and two-qubit gates in a quantum circuit implementation of U applied to an initial state $|\Phi\rangle$ (i.e., $U(\vec{\theta}) = U_k(\theta_k)U_{k-1}(\theta_{k-1})\dots U_0(\theta_0)|\Phi\rangle$). During a VQA run, these parameters are tuned to optimize a cost function $\mathcal{C}(\vec{\theta})$, which in our case is the expectation value of a Hermitian observable $\hat{O}$ relative to the state $|\Psi(\vec{\theta})\rangle$,

$$\mathcal{C}(\vec{\theta}) = \langle \hat{O} \rangle = \langle \Psi(\vec{\theta})|\hat{O}|\Psi(\vec{\theta})\rangle. \tag{5.2}$$

To measure the expectation value of $\hat{O}$ without additional quantum circuitry, it is typical to write $\hat{O}$ as a linear combination of easy-to-measure operators, i.e., Pauli operators $\hat{P}_i \in \{\mathbb{I}, X, Y, Z\}^{\otimes N}$

$$\hat{O} = \sum_i c_i \hat{P}_i \to \mathcal{C}(\vec{\theta}) = \langle \hat{O} \rangle = \sum_i c_i \langle \hat{P}_i \rangle. \tag{5.3}$$

A VQA then passes the estimation of the cost function $\mathcal{C}(\vec{\theta})$ to some classical optimization routine to find the values of $\vec{\theta}$ minimizing $\mathcal{C}$. This optimization loop and the optimizer choices are the focus of this chapter.

To get an estimate of the expectation value $\langle \hat{P}_i \rangle$, one prepares and measures the state multiple times in the $\hat{P}_i$ basis and calculates the mean of the eigenvalues observed. This approximates the cost function $\mathcal{C}$ by an estimator $\bar{\mathcal{C}}$, whose distribution is dependent on the number of repetitions M used to calculate $\langle \hat{P}_i \rangle$,

$$\bar{\mathcal{C}}(\vec{\theta}, M) = \sum_i c_i \left[\langle \hat{P}_i \rangle + \epsilon_i(M) \right]. \tag{5.4}$$

Here, ϵ_i is a random variable drawn from a binomial distribution with variance $\sigma_i^2 \sim 1/M$ that is used to simulate the experimental shot or sampling noise. Assuming that Pauli operators are measured independently, the variance of the estimator $\bar{\mathcal{C}}$ may be propagated directly,

$$\mathrm{Var}[\bar{\mathcal{C}}] = \sum_i c_i^2 \sigma_i^2. \tag{5.5}$$

In general, the assumption of independence is violated. One may measure mutually commuting operators in parallel [124, 125, 132–134, 142]. Then the resulting measurement has non-zero covariance [108, 115], which should be accounted for. However, this only introduces a constant factor to the estimation cost, and will not significantly impact the relative optimzer performance. Here, we will use M, defined in Eq. (5.4), as the overall cost for the quantum subroutine which takes $\vec{\theta}$ and M as inputs, and outputs $\bar{\mathcal{C}}(\vec{\theta}, M)$.

To optimize within a VQA, an access to $\mathcal{C}(\vec{\theta})$ is provided to a classical optimizer, which then minimizes the sampled cost function $\bar{\mathcal{C}}(\vec{\theta}, M)$ as a function of the classical parameters $\vec{\theta}$. One can additionally provide estimates of gradients $\nabla_\theta \mathcal{C}$ (or higher order derivatives) in order to perform gradient-based (or Newton-like) optimization. To avoid a comparison of the runtime of gradient estimation to that of estimating the raw cost function $\bar{\mathcal{C}}(\vec{\theta}, M)$, we only compare four gradient-free optimization algorithms. Moreover, it has been shown that gradient-based optimization strategies suffer given noisy function evaluations with simple noise structures [143] (e.g., stationary and isotropic noisy covariance) in the sense that 1) the convergence rate to local optima is hampered [144] and 2) such simple noise does not help in escaping from local optima [145]. We select the following optimizers (see Appendices for further details):

1. SLSQP determines a local search direction by solving the second-order local approximation of the cost function that satisfies the constrains,

2. COBYLA uses linear approximations of the target and constrains function to optimize a simplex within a trust region of the parameter space,
3. CMA-ES is a population-based optimization algorithm where the points are drawn from a multivariate Gaussian distribution, whose parameters (covariance matrix and location) are adapted online,
4. SPSA employs a stochastic perturbation vector to compute simultaneously an approximate gradient of the objective.

We compare these algorithms across multiple systems of different sizes and number of parameters considered.

5.3. Three-stage sampling adaptation

Existing state-of-the-art quantum hardware is limited by the stability of the devices, which need to be tuned within time-scales of hours up to a day. This influences a hard limit on the total number of samples we can measure before the devices changes, of the order of $\sim 10^9$. This shot budget becomes the limiting factor in VQAs. One must carefully balance between exploring the parameter space and accurately measuring the cost function. In this spirit, Cade et al. [146] introduced a sampling procedure for SPSA that splits the total shot budget between three stages, resulting in improved performance of VQAs.

Naively, one can fix the total number of shots per Pauli operator and run the optimization until the budget is spent (i.e., if one used 1000 shots per Pauli per function call and allocated a total shot budget of 10^7 per Pauli, this would allow for a total of 10000 evaluations). We refer to this approach as one-stage optimization. Alternatively, one could think of an optimization strategy where the number of shots is increased as the optimization progresses towards better parameters as introduced by Cade et al. We perform a three-stage optimization procedure where the number of samples per Pauli operator increases per phase, reducing the number of total function calls. In our three-stage optimization, we fix the number of shots per Pauli for each stage (i.e., 100-1000-10000 shots for a total budget of 10^7). The number of function evaluations is then computed from a ratio 10:3:1. For every 10 function calls at the first stage, we use 3 function calls in the second stage and 1 function call in the third stage (i.e., 7150-2145-715 evaluations for a total budget of 10^7 shots per Pauli).

We compare the one- and three-stage protocols for the four optimization algorithms previously introduced, aimed at assessing if the three-stage protocol has any evident advantage over standard sampling strategy. For this comparison we use the relative energy error,

$$\Delta_r E = \left| \frac{\mathcal{C}(\vec{\theta}_{\text{opt}}) - E_0}{E_0 - c_0} \right|, \tag{5.6}$$

where $\mathcal{C}(\vec{\theta}_{\text{opt}})$ is the noiseless cost function evaluated at the optimized parameters $\vec{\theta}_{\text{opt}}$ obtained from a noisy optimization. E_0 is the lowest eigenvalue of the problem. c_0 is the coefficient of the identity operator, which is the largest Hamiltonian term. This is the relevant figure of merit to capture the performance of the optimizer as it measures the relative error in estimating the traceless part of the Hamiltonian $H - c_0 I$, requiring the quantum computer. Our numerical experiments are performed under sampling noise with a total shot budget of 10^7, 10^8 and 10^9 per Pauli operator. In the one-stage method we fix the total number of function evaluations to 10^4 and use 10^3, 10^4 and 10^5 shots per Pauli operator per function call. In the three-stage procedure the function calls are also fixed at 7150-2145-715 for all budgets, and the shots per Pauli operator at every stage are 10^2-10^3-10^4, 10^3-10^4-10^5, and 10^4-10^5-10^6, respectively. The optimization stops when the shot budget is reached. However, the SLSQP and COBYLA optimization algorithms have a termination criterion that in most cases results in exiting early and not utilizing their full shot budget.

We benchmark the algorithms for three different systems on 8-qubits: H_4 in a chain and square configuration and the 2x2 Hubbard model. The results are shown in figure 5.1. For every problem we run experiments for one-stage (blue, pink and green dots) and three-stage (red, brown, and purple dots) protocols for each optimization. Both SLSQP and COBYLA have a consistent improvement when the three-stage sampling is applied, but are inferior to the performance of CMA-ES and SPSA in all cases. Conversely, for CMA-ES the one-stage sampling has better performance than the three-stage protocol in all three test systems. Finally, SPSA behaves differently with respect to the sampling procedure depending on the system. For the square configuration of H_4 the three-stage sampling has a slightly better energy. In the case of the chain configuration, the one-stage method shows almost a 4-fold improvement over the three-stage method. For the 2x2 Hubbard model the three-stage methods has an order of magnitude improvement in performance. Overall, the best performance across all problems is achieved by SPSA in both one- and three-stage methods (indicated by green and orange ticks on the top of the panel).

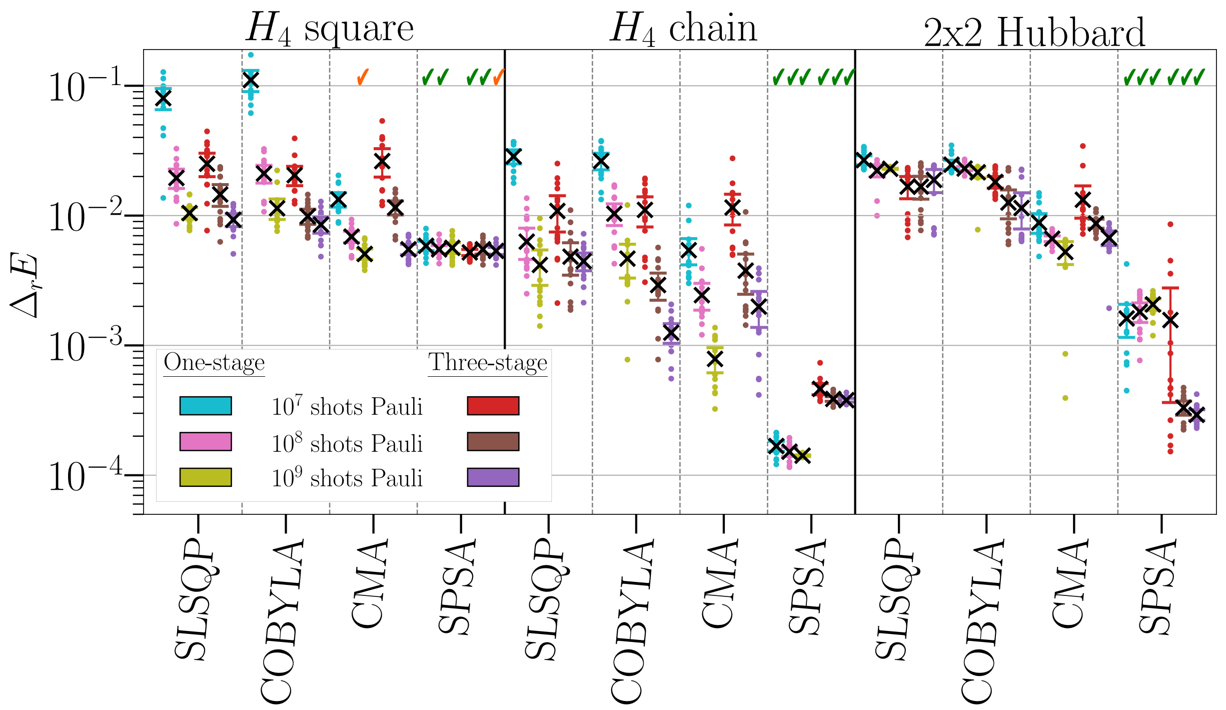

Figure 5.1.: Comparison of optimizers with default hyper-parameters and for the one-stage (blue, pink and green) and three-stage (red, brown and purple) method. Black crosses depict the mean value, the error bars are the 95% confidence interval of 15 independent runs. Green ticks mark the overall winner optimization in mean and standard error. Orange ticks mark the overall winner optimization in mean with overlapping error bars to other optimizers.

5.4. Hyperparameter tuning

Most optimization algorithms come with default (hyper-)parameters detailing the optimization. These (hyper)-parameters are either derived under idealized theoretical assumptions, or evaluated from numerical experiments on standard benchmarks. It is common to tune the hyper-parameters of the optimizers when used on a function that has not been previously studied [147, 148]. Similarly, when performing a VQA one should consider hyper-parameter-tuning the optimizer.

Finding optimal hyper-parameters of an optimizatizer can be costly and generally problem-dependent. A sub-field of classical optimization has been devoted to automatizing such hyper-parameter tuning. Here we use the iterated racing for automatic algorithm configuration [149], IRACE (see Appendix 5.Cfor a description of the procedure), to tune the SPSA and CMA-ES settings for four molecular systems, H_4 square and chain and H_2O at equilibrium and stretched geometries (corresponding to weakly- and strongly-correlated regimes, respectively). Additionally we perform hyper-parameter tuning of CMA-ES for the Hubbard model on three different configurations; 1x6, 2x2 and 2x3. For SPSA, however, we take the results of ref. [146] where the hyper-parameters were tuned. The hyper-parameters used for the numerical experiments can be found in Tables I and II in Appendix 5.C.

With the tuned hyper-parameters we performed a new set of experiments including new systems: H_2O in its equilibrium and stretched geometries with 10 qubits and Hubbard models on 1x6 and 2x3 lattices requiring 12 qubits. The results of these simulations are shown in figure 5.2. A first observation is that CMA-ES improves on the problems tested without tuning. The overall performance between CMA-ES and SPSA is roughly similar, with SPSA doing better in weakly-correlated problems (H_4 chain, H_2O equilibrium and 2x2 Hubbard model). For strongly-correlated systems, CMA-ES has a slightly better performance with the mean values mostly within error bars (orange ticks in fig. 5.2). Supporting our idea that hyper-parameter tuning is crucial to ensure a good VQA performance is the 5-fold improvement of SPSA on the H_4 chain problem. For the H_4 square only a mild improvement is observed for both SPSA and CMA-ES, both reaching an almost equal relative energy error. A reason of this can be that both optimizers find the optimal parameters accessible by the ansatz, better result requiring a larger circuit.

With regards to the Hubbard model, SPSA performs clearly better on the 2x2 configurations. However CMA-ES is capable of finding much lower

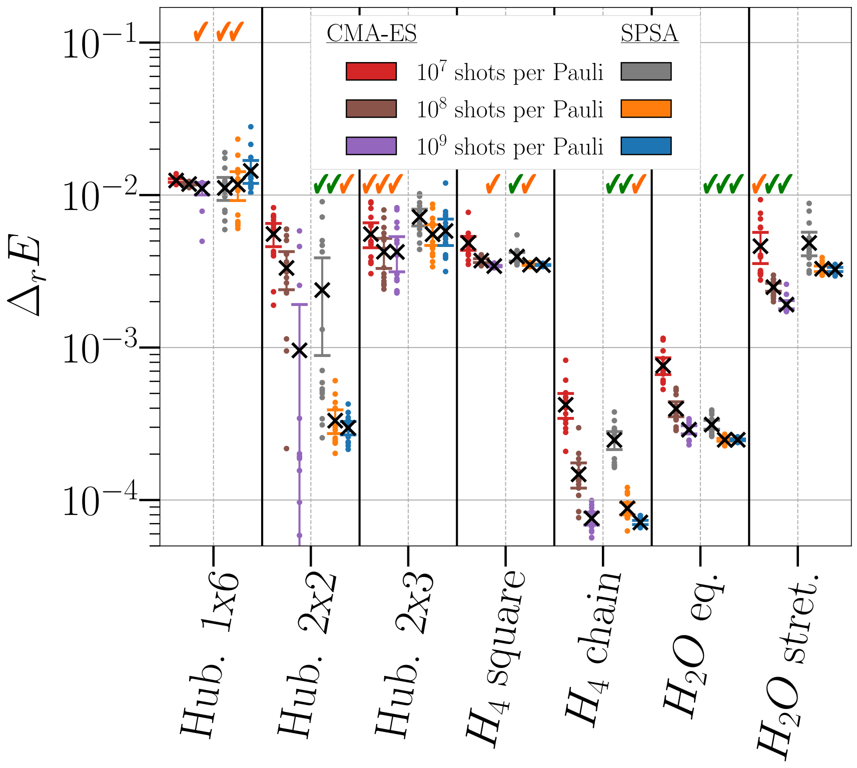

Figure 5.2.: Comparison of optimized hyper-parameters of CMA-ES (red, brown and purple dots) and SPSA (grey, orange and blue dots). Black crosses and error bars depict the mean and 95% confidence interval of 15 independent runs. Green and orange represent the same as in fig. 5.1.

points in with 10^8 and 10^9 optimization, suggesting that the optimization landscape is not trivial, and that better results can be found. The 1x6 and 2x3 Hubbard models yield a comparable performance between the optimizers. Again, a possible explanation for this is that they reach the actual optimum.

Finally, we observe a that CMA-ES starts to outperform SPSA as the system size and number of parameters increases. This might be an indication that SPSA is not as well-suited for large problems.

5.5. The sampling noise floor

A successful VQA requires the optimization algorithm to return the optimal parameters of $\bar{\mathcal{C}}$. It is common to assign as the optimal candidate the one with the best-ever measured $\bar{\mathcal{C}}$. However, in VQAs, the optimization is performed using a proxy cost function $\bar{\mathcal{C}}$ – a sampled version of the real objective $\mathcal{C}$. So, it is possible that $\bar{\mathcal{C}}(\vec{\theta})$ returns a value that is lower than its corresponding noiseless evaluation $\mathcal{C}(\vec{\theta})$ due to statistical fluctuations. Any optimizer that assigns the optimal candidate to the point with the best-ever function evaluation of $\bar{\mathcal{C}}$ will, with large probability, return a candidate worse than the global minimum (assuming its existence). The region of points in parameter space that can return the best-ever measured $\bar{\mathcal{C}}$ we refer to as the sampling noise floor, discussed in the following.

Given a cost function $\mathcal{C}(\vec{\theta})$, we attempt to optimize its sampled version $\bar{\mathcal{C}}$ with variance $\mathrm{Var}[\bar{\mathcal{C}}]$. Let us assume $\mathcal{C}(\vec{\theta})$ has a global minimum $\vec{\theta}_g$ with noiseless value $\mathcal{C}_g$, and that we evaluate $\mathcal{C}(\vec{\theta})$ at multiple $\vec{\theta}$ including one evaluation at $\vec{\theta}_g$. Under sampling noise, the value of the cost function evaluated at $\vec{\theta}_g$ is drawn with some probability $1-p$ from a confidence interval

$$\Delta_p = [\mathcal{C}_g + m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}, -\infty), \tag{5.7}$$

where $m(p) \sim \log(p)$ is the size of the relevant confidence interval for our distribution of $\bar{\mathcal{C}}$. Assuming this distribution is symmetric, whenever $\vec{\omega} \neq \vec{\theta}_g$ satisfies $\mathcal{C}(\vec{\omega}) - m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]} \notin \Delta_p$, with probability $> (1-p)$ an evaluation of the noisy cost function will lie above this confidence region, $\bar{\mathcal{C}}(\vec{\omega}) > \Delta_p$. Then, with confidence $> (1-p)^2$, $\vec{\theta}_g$ will be correctly identified (between these two candidates) as the optimal set of parameters. However, when this is not the case and $\mathcal{C}(\vec{\omega}) - m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]} \in \Delta_p$, we can no longer be confident that the true minimum $\vec{\theta}_g$ will be identified. This defines a (potentially disconnected) region in parameter space,

$$\Omega(p) = \{\vec{\omega} \colon \mathcal{C}(\vec{\omega}) < \mathcal{C}(\vec{\theta}_g) + 2m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}\}, \tag{5.8}$$

from where alternative candidates can be drawn with probability p. This corresponds to a region of possible cost function values,

$$\mathbb{C}_p = \Big[\mathcal{C}(\vec{\theta}_g), \mathcal{C}(\vec{\theta}_g) + 2m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}\Big], \tag{5.9}$$

that an optimizer returning the best-ever measured $\bar{\mathcal{C}}$ may achieve. We define the width of this region, $2m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}$, as the sampling noise floor.

This region is not completely defined, as we have not set a value for p. In practice, the value of p depends upon the rate at which the optimizer can converge; an optimizer that converges slowly will encounter more parameter sets $\vec{\omega}$ near the true minimum, increasing the probability that one of these parameter sets might generate a false optimum. We do not have direct access to an estimate for the width of $\mathbb{C}_p$, but we can still demonstrate the phenomenon numerically.

SPSA and CMA-ES have been designed not to rely on the best-ever function evaluation. In particular, CMA-ES returns two different candidates; the best-ever measured and a so-called favourite. The favourite is computed by the algorithm's update function at the end of the optimization process and includes all accumulated prior information. As this information includes many more shots than a single function call, in principle it can average out the sampling noise over the optimization landscape, and beat the sampling noise floor. We investigate the effect of sampling noise on these two candidates returned by CMA-ES.

Figure 5.3, shows the results of the sampling noise floor in the optimization performance. For every system we computed the following energy values: $\bar{\mathcal{C}}(\vec{\theta}_{\text{best}})$, $\mathcal{C}(\vec{\theta}_{\text{best}})$ and $\mathcal{C}(\vec{\theta}_{\text{fav}})$. For each estimated energy we compute the relative energy error $\Delta E = \frac{\mathcal{C}(\vec{\theta}_{\text{opt}})-E_0}{|E_0-c_0|}$. Note that the best-ever function evaluation (orange points) is often below the true energy, breaking the variational principle. This is due to the effect of sampling noise, and for a fair comparison we should compare $\mathcal{C}(\vec{\theta}_{\text{best}})$ (red points) and $\mathcal{C}(\vec{\theta}_{\text{fav}})$ (purple points), where the true cost function is evaluated. The mean value of $\mathcal{C}(\vec{\theta}_{\text{best}})$ gives an estimate for the width of the sampling noise floor. We observe that in all cases, the mean of the favourite candidate is below the mean of the best candidate, showing that the optimizer has beaten the sampling noise floor. For the Hubbard model and the H_4 systems, this is not significant (up to a 95% confidence interval), but for the two geometries of the water molecule, the difference is much larger. Choosing the favourite over the best yields up to a 3-fold reduction of error. We believe the difference in performance comes from the different optimization landscapes of the different problems; studying this in detail is a target for future work.

5.6. Conclusion

Variational quantum algorithms have recently prompted significant interest as candidates amenable to near-term hardware. However, the per-

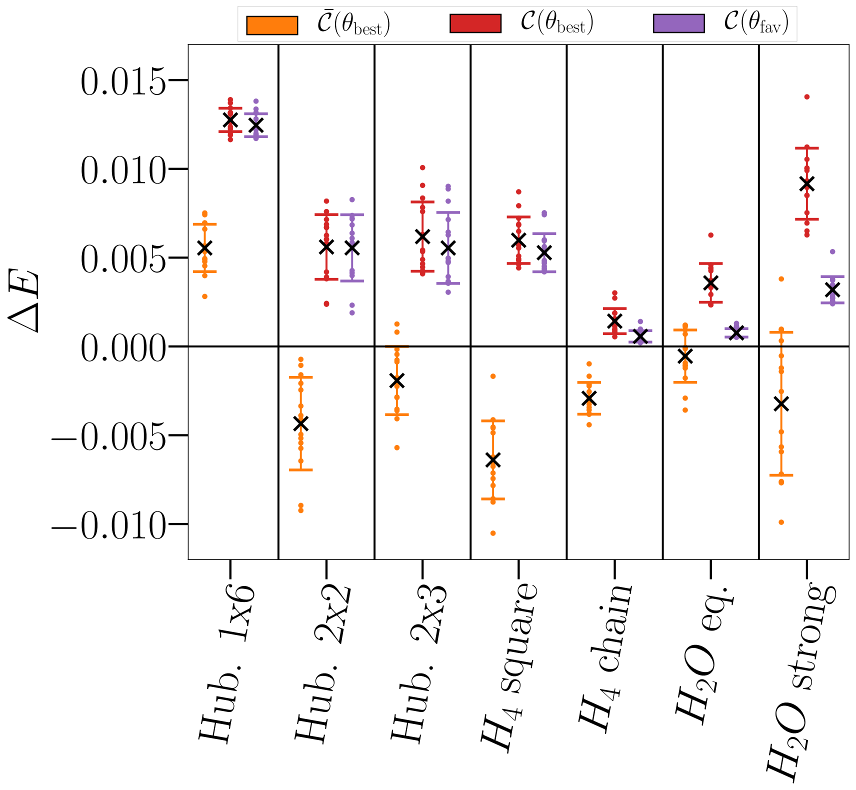

Figure 5.3.: Best-ever versus favourite candidate from CMA-ES under noisy optimization. The optimization uses 10^7 shots per Pauli over the course of the experiment with individual estimations of $\bar{\mathcal{C}}(\vec{\theta})$ are made using only 10^4 shots per Pauli. From left to right: (orange) best-ever measured function evaluation, (red) the best-ever candidate evaluated noiseless, and (purple) favourite candidate evaluted noiseless.

formance of these quantum algorithms relies on a classical optimization of a difficult cost function. This task is in general intractable to solve optimally. It is, therefore, important to benchmark the available optimizers for this purpose. We study the performance of four off-the-shelf optimization algorithms under the effect of sampling noise for the task of finding the ground-state energies. We perform a comparison using default hyper-parameters, and then extend the analysis using a three-stage sampling method from Ref. [146] and by adding hyperparameter tuning. First find that SPSA performs best in both standard and three-stage samplings without tuning. Next, we focus on the performance of SPSA and CMA-ES in the three-stage procedure. We then hyper-parameter-optimize these two methods. With these new parameter settings, SPSA and CMA-ES have a comparable performance on the strongly-correlated systems and a small advantage for SPSA on the weakly-correlated ones. We notice that the advantages of SPSA seem to vanish as problem sizes grow.

Finally, we study the effect of sampling noise on the optimization performance using CMA-ES. We observe that the best-ever function evaluation is may not be a feasible optimal candidate, contrary to the common approach in classical optimization. Specifically, we show that the best-ever result suffers from a sampling noise floor problem that makes any of the parameters within it a potential best-ever result. In contrast, the so-called CMA-ES favourite candidate obtained from its update rule at the end of the optimization shows an overall better mean and standard deviation than its best-ever counter-part, indicating the sampling noise floor can be overcome. We expect that our analytical and numerical results of the sampling noise floor opens a new line of inquiry about optimization methods for VQAs.

Appendix

5.A. Appendix: Details on optimization algorithms

In this appendix, we provide a more detailed description of the optimization algorithms used in this chapter.

- Simultaneous perturbation stochastic approximation algorithm [150, 151] (SPSA) is designed for noisy evaluations of a cost function, where a stochastic perturbation vector (for instance, a vector whose components are independently sampled from the Rademacher distribution) is used to simultaneously estimate all partial derivatives at given a point. Compared to the well-known finite difference method to estimate the gradient, which requires $2d$ evaluations of the cost function defined over $\mathbb{R}^d$, the stochastic approximation always consumes two evaluations, hence saving many function evaluations when the search dimension is high. However, this algorithm does not follow exactly the gradient direction due to the use of stochastic perturbation.

- Constrained Optimization BY Linear Approximations (COBYLA) [152] is designed for constrained derivative-free optimization. It employs linear approximations to the objective and constraint functions via a linear interpolation given $M + 1$ points (or simplex). These approximations are then optimized within a trust region at each step.

- Sequential Least Squares Quadratic Programming (SLSQP) [153, 154] is an implementation[1] of the more general Sequential Quadratic Programming (SQP) approach [155] for solving constrained optimization problems. Loosely speaking, in each iteration, SQP proposes a local search direction by solving a sub-problem defined at

[1] We took the implementation from the `scipy` package, which is based on the original software as described in [153].

the current search point in which the nonlinear cost function is replaced by its local second-order approximation and the constraints are approximated by their affine approximation. When there is no constraint, this method degenerates to Newton's method.

- Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [156] is the state-of-the-art direct search algorithm for continuous black-box optimization problem, which distinguishes itself from other algorithms in self-adaptation of its internal variables to the energy landscape. Briefly, this algorithm iteratively draws a number of candidate solutions from a multivariate Gaussian distribution, in which the shape of this distribution (e.g., covariance matrix and location) is adapted online based on the evaluated points in its trajectory.

5.B. Appendix: Numerical experiments

In this appendix, we describe the numerical experiments used to generate the data for the figures of the chapter. The code and data to reproduce these figures can be found in [157].

To generate the target problems we use the open-source electronic structure package OpenFermion [90]. In addition, we generate the molecular systems with the computational chemistry software Psi4 through the OpenFermion plug-in. The classical numerical simulations are performed using the open-source quantum circuit simulator package Cirq [158]. Regarding the optimization methods we use the Scipy [159] sofware for COBYLA and SLSQP, PyCMA [160] for CMA-ES and an in-house version of SPSA based on the code in [161].

As described in the main text, we focus on the performance of the optimization methods for VQAs under sampling noise conditions. In order to include the sampling noise in our experiments we compute a noisy expectation value for every Pauli operator in the Hamiltonian with a fixed number of shots, as follows:

1. Prepare the ideal quantum state, measure $\langle P_i \rangle$ and $p = \frac{1-\langle P_i \rangle}{2}$,
2. sample $\tilde{p} = \mathcal{B}(p, M)$ from a binomial distribution with M shots,
3. compute a noise expectation value $\langle \tilde{P}_i \rangle = 1 - 2\tilde{p}$,
4. calculate the noisy Hamiltonian expectation value as

$$\langle \tilde{H} \rangle = \sum_i c_i \langle \tilde{P}_i \rangle. \tag{5.10}$$

H_2O	Equilibrium	Stretched
O	(0.0, 0.0, 0.1173)	(0.0, 0.0, 0.0)
H	(0.0, 0.7572, -0.4692)	(0.0, 1.8186, 1.4081)
H	(0.0, -0.7572, -0.4692)	(0.0, -1.8186, 1.4081)

Table 5.1.: Table describing the configurations of the atoms for the two water molecule problems used in this chapter.

This is a good approximation to the sampling noise generated by measuring the expectation values of Pauli operators in real hardware, when the number of shots is large enough. Moreover, we avoid the bottleneck of preparing and measuring the same state multiple times.

In the Fermi-Hubbard model experiments (see app. 5.E for further details), we set the parameters of the Hamiltonian to $t = 1.0$ and $U = 2.0$. The ansatz circuit for these problems is constructed using the Variational Hamiltonian Ansatze (VHA) with 5 layer for the 1x6, 2 layer for the 2x2 and 4 layers for the 2x3 Hubbard model. These are the minimum number of layers needed to achieve a ground-state fidelity of 0.99 in ref. [146].

For the H_4 in the chain configuration, the first hydrogen atom is located at 0.0 in all coordinates, then every atom is separated in the x-direction by 1.5Å. In the square configuration, we fix the hydrogen atoms in 2-dimensions. The positions of the atoms are parametrized by their polar coordinates with $R = 1.5$Å and $\theta = \frac{\pi}{4}$, and we locate them at $(x, y, 0), (x, -y, 0), (-x, y, 0), (-x, -y, 0)$ with $x = R\cos(\theta)$ and $y = R\sin(\theta)$. For the water molecule problems, the (x, y, z)-coordinates of the atoms given in Table 5.1. Additionally, in both of the problems we reduce the active space by freezing the lowest two lowest orbitals, thus reducing the problem from 14 qubits to 10 qubits (or from 7 to 5 spin-orbitals). As a trial state to approximate the ground-state of the molecular systems, we use the so-called Unitary Couple-Cluster ansatze. A detailed description on how we construct the UCC ansatze can be found in a separate appendix 5.D.

Finally, the total number of parameters to be optimized for each target problem can be found in table 5.2.

System	# Parameters
H_4 chain	14
H_4 square	10
H_2O eq.	26
H_2O stret.	26
Hub. 1x6	15
Hub. 2x2	6
Hub. 2x3	16

Table 5.2.: Number of parameters of the ansatze for each target problem.

5.C. Optimization algorithms hyper-parameters

Prior to applying the aforementioned optimizers on VQAs, we also optimize the hyper-parameters of those optimizers. Such an extra tuning task aims at bringing up the performance of each optimizer to the maximum, hence facilitating a fair comparison on each problem. To achieve this task efficiently, we utilize the well-known IRACE algorithm f for the hyperparameter tuning. Irace has been extensively applied in automated machine learning researches for configuring machine learning models/optimizers [162, 163].

Built upon a so-called iterated racing procedure, this algorithm employ a statistical test (usually the Wilcoxon ranked-sum test) to obtain a robust (with respect to the sampling noise in measured energy values) ranking of hyper-parameter settings, thereby serving as a suitable choosing for our task. The tuning process with IRACE initiates by fixing the subset of optimizer hyper-parameters to be modified, including bounds and potential constrains. Then, a 'race' between randomly sampled values begins. These configurations are evaluated a fixed number of times, and the less favourable configurations are disregarded based on a statistical test. The configurations that survived are then raced again until the budget of evaluations is depleted or the number of configurations is below a threshold. Next, IRACE updates the candidate generation model based on the survival configurations, and generates a set of new configurations to race against the elites. The racing procedure is repeated until the total budget is depleted. The surviving configurations are returned as the optimal configurations of the algorithm. The final hyper-parameters used for the experiments are the average of the survivors are shown in Table 5.3

SPSA	a	α	c	γ
default	0.15	0.602	0.2	0.101
H_4 chain	1.556	0.809	0.106	0.097
H_4 square	0.867	0.593	0.133	0.113
H_2O eq.	0.103	0.878	0.149	0.131
H_2O stret.	0.660	0.743	0.253	0.108
Hubbard	0.15	0.602	0.2	0.101

Table 5.3.: List of values for SPSA hyper-parameters used in Fig. 5.2 after tuning using IRACE. We perform hyper-parameter optimization only with H_4 chain and H_2O equilibrium and use the same values for the respective square and stretched configurations. For the Hubbard models we take the default values as ref. [146] suggest their optimality.

and 5.4.

In detail, the hyper-parameters we tuned are as follows:

- For SPSA, we use the following ranges for each hyper-parameter: $a \in [0.01, 2]$, $\alpha \in [0, 1]$, $c \in [0.01, 2]$, and $\gamma \in [0, 1/6]$.
- For CMA-ES, we use the following ones: population size $\in [30, 130]$, c mean $\in [0, 1]$, $\mu \in [0, 0.5]$, Damp. factor $\in [0, 1]$, and $\sigma_0 \in [0.25, 1.1]$.

For running the irace algorithm, we allocated 500 evaluations of the hyper-parameters as the total budget, as well as a maximum total running time of 7 days, and used two evaluations of each hyper-parameter in the beginning of each race. Also, we used the F-test for eliminating worse configurations in the racing procedure. The finally suggested configurations in Table 5.3 and 5.4 are the best elites from four independent runs of irace.

As for COBYLA and SLSQP, we took their default hyper-parameter settings, i.e., $\rho_{\text{initial}} = 0.1$ and Tolerance$= 10^{-8}$ for COBYLA and $\epsilon = 0.055$ and Tolerance$= 10^{-8}$.

5.D. Appendix: Unitary Coupled-Cluster ansatz based on coupled-cluster amplitudes

Several classes of systems remain challenging to solve, even for the coupled cluster methods considered as the golden standard in quantum chemistry.

CMA-ES[2]	σ_0	Population	μ	c mean	Damp. Factor
default	0.15	$\lceil 4 + 3\log(m) \rceil$	0.5	1.0	1.0
H_4 chain	0.20	149	0.383	0.293	0.665
H_4 square	0.309	99	0.409	0.561	0.852
H_2O eq.	0.344	99	0.460	0.192	0.770
H_2O stret.	0.310	104	0.380	0.802	0.819
Hub. 1x6	0.9131	51	0.3814	0.3614	0.6006
Hub. 2x2	0.8561	113	0.2741	0.6317	0.6771
Hub. 2x3	0.897	128	0.1898	0.988	0.8391

Table 5.4.: List of values for CMA-ES hyper-parameters used in Fig. 5.2 after tuning using IRACE. Here, m indicates the number of free parameters of the ansatz in each problem.

Those systems are usually plagues by “quasidegeneracy”, meaning that the wavefunction cannot be decomposed into a single leading component. This leads to an important deterioration of methods relying on the single determinant assumption (also said to be mono-reference) [164]. This issue can be partially solved by developing multi-reference coupled cluster approaches (see Refs. [165, 166] for a review). Owing to the recent developments of quantum algorithms in the NISQ-era, there has been a renewed interest in the unitary formulation of coupled cluster (UCC) which is naturally suited for quantum computation and naturally extendable to generate multi-reference wavefunctions [39, 167], while being intractable on classical computers [115]. Several formulations of UCC have been investigated to go beyond the standard UCCSD method where only fermionic excitations from occupied to virtual orbitals (with respect to the reference determinant, usually the Hartree–Fock one) are considered [115, 168–172]. However, the number of operators (and thus the number of parameters) can rapidly become problematic if implemented naively. A powerful approach is provided by the Adaptive Derivative-Assembled Pseudo-Trotter (ADAPT) types of ansatz [172–178], which allows to adaptively increase the number of operators in the ansatz one by one until reaching a given accuracy. In this chapter, we employ a different strategy by taking advantage of the amplitudes extracted from the traditional coupled cluster method performed on a classical computer. In coupled cluster, the exponential ansatz reads as follows

$$|\Psi(\vec{t})\rangle = e^{\hat{\mathcal{T}}}|\Phi_0\rangle, \tag{5.11}$$

where $|\Phi_0\rangle$ denotes the reference determinant (like the Hartree–Fock wavefunction) and

$$\hat{\mathcal{T}} = \sum_{i=1}^{\eta} \hat{\mathcal{T}}_i = \sum_{\mu} t_\mu \hat{\tau}_\mu \tag{5.12}$$

(η denotes the total number of electrons) is usually truncated to singles and doubles only:

$$\begin{aligned} \hat{\mathcal{T}}_1 &= \sum_{\substack{i \in \text{occ} \\ a \in \text{virt}}} t_a^i \hat{a}_a^\dagger a_i, \\ \hat{\mathcal{T}}_2 &= \sum_{\substack{i > j \in \text{occ} \\ a > b \in \text{virt}}} t_{ab}^{ij} \hat{a}_a^\dagger \hat{a}_b^\dagger a_i a_j. \end{aligned} \tag{5.13}$$

One could think of determining the CC amplitudes $\mathbf{t}$ variationally, but this is not convenient in practice because the Baker–Campbell–Hausdorff (BCH) expansion cannot be used (because $\hat{\mathcal{T}}^\dagger \neq -\hat{\mathcal{T}}$). Tractable implementations rely on a non-variational optimization using the "Linked" formulation:

$$e^{-\hat{\mathcal{T}}} \hat{H} e^{\hat{\mathcal{T}}} |\Phi_0\rangle = E(\vec{t})|\Phi_0\rangle. \tag{5.14}$$

The amplitudes are then determined by solving a set of non-linear equations defined by projecting Eq. (5.14) against a set of excited configurations $\{|\mu\rangle\}$ (configurations obtained from the excitation operators in $\hat{\mathcal{T}}$):

$$\langle \mu | e^{-\hat{\mathcal{T}}} \hat{H} e^{\hat{\mathcal{T}}} |\Phi_0\rangle = 0, \tag{5.15}$$

for which the BCH expansion can be used, as it can be naturally truncated to fourth order.

In this chapter, we computed the coupled cluster amplitudes of all our molecular systems (H_4 chain, H_4 square and H_2O) and defined our UCC ansatz according to these amplitudes. Instead of implementing UCC naively by considering all possible excitations, we only keep the excitation operators for which the corresponding CC amplitude is non-zero. This reduces already the total number of operators (and thus the total number of parameters) significantly. In practice, we use the trotterized-UCC ansatz,

$$|\Psi(\vec{\theta})\rangle = \prod_{\mu} e^{\theta_\mu (\hat{\tau}_\mu - \hat{\tau}_\mu^\dagger)} |\Phi_0\rangle. \tag{5.16}$$

This trotterized form is an approximation (though it may be mitigated by the classical optimization [179]) which depends on the ordering of the operators. We decided to order the operators with respect to the value of the CC amplitudes in descending order, meaning that the first operator to be applied to the reference state within the UCC ansatz will be the operator with the highest associated CC amplitude. We figured out that the operators in our ansatz were also the ones picked by the ADAPT-VQE ansatz [172], although the ordering might not different. However, ADAPT-VQE can add new operators (or select and repeat an already present operator) to reach a higher accuracy. To avoid performing the (somewhat costly) first ADAPT-VQE steps, one could think about using our strategy first and then apply ADAPT-VQE for few more steps to increase the pool of operators slightly. Note that a stochastic classical UCC can also be employed as a pre-processing step to determine the important excitation operators of the UCC ansatz, as shown in the recent work of Filip *e*t al. [180].

In our numerical experiments the initial state is always the Hartree–Fock state corresponding to the number of electrons in the system. The parameters of the circuit are initialized at 0.0.

5.E. Appendix: Variational Hamiltonian ansatz for the Hubbard model

In this section, we provide the details on the variational Hamiltonian ansatze (VHA) used for the Hubbard model problems.

The Fermi-Hubbard Hamiltonian describes the behaviour of fermions on a lattice of n_x x n_y sites. Fermions can hope to nearest-neighbour sites with some strength t, and observe a repulsion or Coulomb term of strength U to move to the same site with the same spin,

$$H_{\text{Hubbard}} = H_{\text{t}} + H_{\text{U}} = \\ -t \sum_{(i,j),\sigma} \left(a^\dagger_{i\sigma} a_{j\sigma} + a^\dagger_{j\sigma} a_{i\sigma} \right) + U \sum_i n_{i\uparrow} n_{i\downarrow}.$$

One can further split the hopping term with respect to the vertical and horizontal hopping terms $H_{\text{t}} = H_v + H_h$.

The VHA were introduced in ref. [119] as a means of constructing parametrized quantum states motivated by time-evolution by Troterrization for the Hubbard model. However, in our numerical experiments we use the VHA introduced by Cade et al. [146] where the horizontal and vertical terms can be implemented in parallel (see eq.[2] in reference). The parametrized quantum state is constructed as

$$|\Psi(\vec{\theta})\rangle = U(\vec{\theta})|\Phi\rangle = \Pi_{l=1}^{L} e^{i\theta_{v_2,l}H_{v_2}} e^{i\theta_{h_2,l}H_{h_2}} \tag{5.17}$$

$$e^{i\theta_{v_1,l}H_{v_1}} e^{i\theta_{h_1,l}H_{h_1}} e^{i\theta_{U,l}H_U}. \tag{5.18}$$

The initial state $|\Phi\rangle$ is the Gaussian state of the non-interacting part of the Hamiltonian, and the parameters of the circuit are set to 0.0.

The Fermi-Hubbard Hamiltonian are generated with the open-source package OpenFermion [90].